

JAG - EN RUP-MUPP

Magnus Mickelsson

Min kontakt med Rational Unified Process (RUP) har gått igenom ett antal faser genom åren; från initialt imponerad till bitter och frustrerad, och vidare till pragmatisk. Det vill säga på samma sätt som med många andra tekniker, verktyg och metoder.

Det började för ett antal år sedan med ett storslaget RUP-införande i en organisation, där ingen i teamet var bekant med konceptet utom de speciella metod-gurus som var inkallade för att se till att RUP efterlevdes.

I uppdraget ingick det att producera mängder av dokument, för att passa in med de artefakter som RUP definierade. Att ha dessa artefakter var tydligen nyckeln till att bedriva ett framgångsrikt projekt!

Nu är det säkert så att en del av er skadeglatt nickar igenkännande, en del skakar sorgset på huvudet och en del vet inte riktigt vad de andra har för anledning att motionera nackmuskeln överhuvudtaget. Det kommer uppenbara sig.

I projektet skapades så pass mycket dokument att vi till slut fick problem med att hinna/orka underhålla dem vid sidan av allt annat som skulle göras. Dokumenten blev därför ofta lagda i malpåse och sällan använda. Mycket av materialet var alltså rent *slöseri*.

De resultat som beställarna **verkligen** behövde drunknade i alla andra saker som producerades, blev försenade och inte riktigt på det vis som de hade tänkt sig. Det är lätt att bli vilseledd från det som trots allt är det viktigaste med att producera IT-stöd, att leverera detta stöd till någon så det kommer till användning.

“Under pressure, even the best of project managers fall back to {Waterfall} practices. That's the bugbear we all fear here.”

- Citat från c2.com

En annan högst bidragande orsak till

problemen var mängder med tekniskt strul. Detta framkom framförallt en bit in i

projektet, eftersom vi inte hade doppat tårna i vattnet djupt nog initialt, utan fokuserat på en detaljplan, noggranna use-cases och väl genomtänkta arkitekturskisser.

Efter hand, då frustrationen ökade, gav jag mig in i att försöka förstå varför inte RUP, som vi använde den, var en lösning som fungerade i praktiken. I mitt sökande stötte jag på en artikel skriven av Craig Larman (Valtech), Philippe Kruchten och Kurt Bittner (Rational); "How to fail with the Rational Unified Process - seven steps to pain and suffering".

Stegen till lidande är som följer:

- Applicera vattenfallsmetoden på RUP (skyldig - till exempel producerades en mycket detaljerad kravspecifikation)
- Använd RUP som en tung, förutsägbar process (skyldig - vi hade en lång planeringsfas för att utreda hur lång tid projektet skulle ta, där projektledaren sedan lade på en lämplig marginal för att estimaten skulle vara rimliga)
- Undvik kunskaper om Objekt-Orientering (oskyldig, tro det eller ej!)
- Undervärdera **anpassningsbar**, iterativ utveckling (skyldig - uttrycket "nu fryser vi kraven" förekom en hel del från styrgrupps- och projektledningshall)
- Undvik mentorer som förstår iterativ utveckling (skyldig - en av våra mentorer läste artikeln av Larman/Kruchten/Bittner och sade "oj, jag inser själv att jag haft en del vattenfallstänk")
- Anpassa RUP som en big-bang aktivitet (skyldig - det var inte en gradvis anpassning utan pang på)
- Ta råd från felinformerade källor (skyldig - mentorerna i vårt fall, samt leverantörerna av programvaran som strulade)

Jag lärde mig oerhört mycket från projektet, och har sedan dess fortsatt med att gräva i litteratur och prata med kollegor kring hur vi egentligen jobbar i våra projekt. Det jag kommit fram till är att det är oerhört vanligt att begå samma misstag som vi gjorde.

Även för dem som oftast arbetar lättroligt händer det att projektet börjar tänka i vattenfallstermer, speciellt när det börjar bli bråttom att nå ett releasedatum. Kvalitet slängs åt sidan för att kunna snabbt leverera något som kanske är lite skakigt, men ändå går att använda.

Det tråkiga är att oftast kommer aldrig kvalitet med på dagordningen igen, på grund av att projektet hamnar i en neråtgående spiral av hets → brist på kvalitet → ineffektivitet/buggar/problem → ännu mer hets och så vidare.

Kortsiktiga lösningar leder oftast till problem. Om dessa problem inte åtgärdas så snart som möjligt blir kvalitetsskulden större och större, koden blir onödigt komplex, klumpig och full av buggar, vilket gör det svårt att hitta fel och lägga till ny funktionalitet. Lasten på projektet blir tyngre och tyngre, något ofta inte projektledare kan förstå följderna av - framförallt om de inte själva har utvecklarbakgrund. Det enda som syns i lägesrapporterna är att tidsplanerna inte hålls, av okänd anledning.

Symptomen ser ut som att teamet arbetar och arbetar, men hinner inte leverera speciellt mycket. *Sjukdomen* är ofta en ineffektiv process och en stor kvalitetsskuld i plattformen, men beror oftast också på andra faktorer. Ett exempel kan vara att utvecklarna inte är vana att tänka i termer av kvalitet, eller har en felaktig uppfattning om vad kvalitet innebär. Tyvärr försöker många åtgärda symptomen, på mängder av olika sätt, och inte själva sjukdomen.

En liknelse är att projektet sitter i en bil och försöker ta sig från punkt A till punkt B. Ingen har brytt sig om att bilen behöver service, olja och bensin utan kör på så det (bokstavligen) ryker.

När bilen börjar få problem försöker man lösa detta genom att anställa en kontrollant för att se till att föraren sköter sitt jobb, eller anställer ett antal personer att putta på bilen för att ta sig framåt. Detta i stället för att se till att stanna vid en mack och ge bilen den service den så desperat behöver.

På kortare sikt är det bättre att anställa bilputtare (2-3 kilometer kanske), men för att komma till resans mål går det mycket snabbare att spendera en dag på verkstaden innan man åker vidare.

När jag via Citerus kom i kontakt med Scrum, kändes det som jag hade hittat ett bättre sätt att slutföra resan på.

Varför, kan man undra?

- Fokus ligger på att producera högkvalitativ mjukvara som levererar nytta för en beställare.
- Scrum implementerar till viss del filosofin kring *Lean*, Toyotas process för produktutveckling och produktion. Idén är att trimma sina processer för att det ska gå snabbt utan att kvalitén glöms bort; det är i stället en förutsättning för att kunna arbeta effektivt över huvud taget.
- Processen är anpassad för en föränderlig värld och utvecklas med omständigheterna. Efter varje sprint sätter sig teamet ner och funderar på hur

arbetet kan flyta på bättre. Detta garanterar att teamet lär sig vad som fungerar och inte, och får chansen att införa förbättringar regelbundet.

- Det ingår i Scrum att problem hela tiden kommer upp till ytan och måste tas om hand å det snaraste, inte skjutas upp på obestämd tid eller tappas bort på vägen. Projektarbetet fungerar då väl både på kortare och längre sikt, eftersom effektiviteten i det dagliga arbetet är högre då det inte ligger konstanta hinder i vägen.
- Alla får den information de behöver kring projektets framsteg och problem, vilket gör att ett mått av *verklig, och inte inbillad* kontroll finns, som ett verktyg att ta fram då det behövs.
- Beslut fattas av dem som har mest information. Projektledare ska inte fatta beslut bara på basis att de är projektledare, utan i stället se till att teamet som sitter på detaljkunskaperna verkligen förstår marknadens/kundens behov och arbetar på ett effektivt, kvalitativt sätt.
- Uppgifter att lösa delas inte ut av projektledaren via detaljstyrning, de **tas** i stället av medlemmarna i teamet. Det Tayloristiska sättet - det vill säga att projektledaren detaljstyr, bygger i grunden på uppfattningen att projektmedlemmarna är lata, oansvariga och kan inte själva förstå vad som är bäst för projektet, något som hängt med från tiden då industrialismen slog igenom. Motivation och kreativitet dämpas, vilket innebär att produktiviteten minskar. Scrums sätt innebär att de som **vill** och kan ta en uppgift, löser den. Gärna i samarbete med andra i teamet, för att dela kunskaper och minska risken för fel.
- Beställarna får hela tiden se mjukvaran växa fram framför deras ögon, vilket bygger förståelse och förtroende, och leder till att det i slutändan är ett system som beställarna verkligen vill ha som levereras - inte något som beslutades när det fanns minimalt med information tillgänglig.
- Scrum utger sig inte för att vara en process som löser alla problem. Tvärtom för Scrum över ansvaret för att lyckas på ett team av samarbetande, kompetenta och konstant lärande individer, med olika kunskaper och erfarenheter.

Hur har då Scrum att göra med om jag numera är en RUP-mupp eller inte?

Jo, jag har lärt mig ett sätt att arbeta enligt RUP som jag själv trivs med. Ursprunget till vad jag hittade finns dels definierat i RUP självt, och dels i en presentation jag snubblade över på nätet: *Comparing/combining RUP, XP, and Scrum - mixing the process cocktail*.

Rational rekommenderar nämligen att RUP anpassas till den faktiska situationen, något även mentorerna i mitt tidigare uppdrag arbetade med. Presentationen jag hittade lyfte fram bra exempel på hur tankegångarna kan se ut vid en sådan anpassning, något som lett till att jag nu bättre kan hitta en utgångspunkt för en metod som är anpassad till problemställningen.

Jag tar det som fungerar bäst från Agile-metoderna Scrum och XP, blandar sedan i en smula RUP, allt efter smak och behov. Jag håller idéerna från Lean Software Development i bakhuvudet, och försöker eliminera slöseri från processen om det går.

Resultatet blir en metod som hänger med i en föränderlig värld, som är lagom lätttrölig och framförallt som fokuserar på det som trots allt är viktigast; att leverera kvalitativ mjukvara till beställaren så snart som möjligt.

Så, om ni hängt med så här långt är svaret på frågan ”Jag - en RUP-mupp?”... Jajjemen! (När metoden används på ett bra sätt!)

Hoppas ni fick ut något av resan från frågan till svaret, det är ju trots allt oftast själva resan som ger mest i slutändan. ■ ■ ■

LÄSNING FÖR ATT BLI EN TVÄTTÄKTA RUP-MUPP

- How to fail with the Rational Unified Process - seven steps to pain and suffering
<http://www.agilealliance.org/articles/larmancraigkruchtenph/file>
- Waterfall Method (för att känna igen symptomen av vattenfallstankar i processen)
<http://www.lshift.net/waterfall.html>
- Comparing/combining RUP, XP, and Scrum - mixing the process cocktail
http://www.netobjectives.com/events/download/rup_xp_scrum_pc_030326_ppt.pdf
- RUP in the dialogue with Scrum
<http://www-128.ibm.com/developerworks/rational/library/feb05/krebs/>
- Poppendieck, Tom & Mary: Lean Software Development: An Agile Toolkit, ISBN 0-321-15078-3
<http://www.citerus.se/shop>



Magnus Mickelsson är konsult på Citerus AB och arbetar som systemarkitekt och utvecklare. Han har ett brinnande intresse för hur man kan förbättra sin utvecklingsprocess, både ur ett verktygs- och metodperspektiv. Kontakta honom på magnus.mickelsson@citerus.se om du vill prata om hur du kan undvika att få en vattenfallsmetod när du inför RUP.